AMENDMENTS TO THE SPECIFICATION

Please replace the paragraph beginning on page 1, line 7, with the following rewritten paragraph:

-- Data security can be compromised by exploiting aspects of the underlying network protocols used for ~~transport. Specifically,~~ transport, specifically, aspects related to the ~~protocols~~ processing and reassembly of data at end host destinations. Protocols such as TCP, IP, UDP, etc. specify how a datagram (i.e., packet, fragment, segment, stream, etc.) is handled during data communication. Variations in transmission, reception, and interpretation may result in different interpretations of reassembled data received at an end host and another inline device or application (e.g., firewall, network security management, intrusion detection, vulnerability assessment, etc.). Protocols may be manipulated in such a manner that this ambiguity can be used to hide the presence of a threat from a security device.--

Please replace the paragraph beginning on page 3, line 15, with the following rewritten paragraph:

-- Figure 6B illustrates a further process for fragment reassembly, in accordance with an embodiment; [[and]]--

Please replace the paragraph beginning on page 4, line 1, with the following rewritten paragraph:

-- Figure 7 illustrates a process used in an embodiment to process overlapping fragments, as in step 612 of Figure 6B; and[[.]]--

Please replace the paragraph beginning on page 5, line 20, with the following rewritten paragraph:

--Figure 1 illustrates a system for efficient assembly of fragmented network traffic for data security. System 100 shows an attacker 102 transmitting data across a network 104, which may include one or more nodes (as illustrated), sending data to an inline security device such as firewall 106. In other examples, firewall 106 may be replaced by other types of inline devices or applications for security management, intrusion detection, network security, etc. In still other embodiments, firewall 106 may be combined with an end host 108 in an inline configuration. Downstream of firewall 106, an end host 108 and a destination host [[116]] 118 are present. In this example, end host 108 may be a computer, server, router, switch, or other type of computing device having an installed security application or system (e.g., network security, security management, intrusion detection, vulnerability assessment, etc.). End host 108 has an installed security application, such as those described above, while destination host [[116]] 118 is the intended recipient of the data traffic stream.--

Please replace the paragraph beginning on page 6, line 19, with the following rewritten paragraph:

-- As datagrams are received by host 108, information contained in the datagram of the packets are evaluated to determine whether to reassemble, hold, process, or discard the packets. Prior to this evaluation, the headers of the datagram are used to assemble the data into the appropriate order 116. If fragmented datagrams are received, the host 108 decides how to assemble them. If ambiguities exist in the protocol used (e.g. IP), then the resulting assembled data 114 may have several possible variations. For example, and as explained in more detail below in connection with Figure 2, if consecutive data fragments are sent with an overlapping offset, depending on the protocol used and the configuration of the host receiving the fragments (e.g., operating system, etc.) the fragments may be reassembled in different ways, as illustrated by the two options shown in assembled data 114. If such an ambiguity exists and the host 108

resolves the ambiguity differently than the asset host 118, the host 108 may reassemble the fragmented datagrams differently than the asset host 118, with the result that the [[NIDS]] network intrusion detection system (NIDS) or other security process running on the host 108 may see different data going to the asset host 108 than the asset host 108 actually received. In this example, the data 116 as reassembled at host 108 is compared to known invalid forms of data in the signature table 110. If a match occurs, then the packets are invalidated and identified for further processing (e.g., discard, hold, etc.). If not, then the packets will continue to be processed normally. Similarly host 118 also has a flow table 120 for reassembling received data packets, fragments, segments, etc. Host 118 processes the incoming datagram and inserts it into the data pool for the flow. Datagram headers identify where to insert the particular datagram into the data pool. In the cases of a fragmented datagram, the host 118 applies a local policy to determine how to process and insert the fragments. As noted above, if the local policy for reassembling fragmented datagrams applied by host 118 is different than the corresponding policy applied by the host 108, e.g., because they have different operating systems and/or are otherwise configured differently, the result may be that the NIDS or other security process running on host 108 may see in the flow associated with asset host 118 data matching an attack signature when in fact the data as received by asset host 118 is innocuous, resulting in a false positive result at the NIDS, or the NIDS may see in the flow only innocuous data when in fact the data as received by asset host 118 matches an attack signature that would otherwise have been detected by the NIDS.--

Please replace the paragraph beginning on page 10, line 13, with the following rewritten paragraph:

-- Figure 4 illustrates a process (400) for monitoring network traffic for data security, in accordance with an embodiment. One or more packets are received, for example, from an

attacker 102 (402). As datagrams are received, a data traffic stream is re-assembled (404). As the data traffic stream is reassembled, packets are scanned for attacks and threats (406). Scanning the data traffic stream for anomalies, as described above, may be used to determine whether an attack or threat has occurred or is in progress (408). If an attack or threat is detected, then the data traffic stream may be blocked from reaching destination 108, if the security system or application is so configured, and/or an alert may be sent to a network security manager or some other recipient or system (410). If no attack or threat is detected, then the packets are ignored and, if necessary, forwarded to their destination (412).--

Please replace the paragraph beginning on page 11, line 3, with the following rewritten paragraph:

-- Figure 5[[A]] illustrates a process for reassembling a data stream, as in step 404 of Figure 4, in accordance with an embodiment. In this embodiment, a datagram is decoded during reassembly (502). Once decoded, the datagram may be examined to resolve the data into a header and payload (504). The datagram and its respective header and payload are examined further to determine whether the datagram is a fragment of a larger data traffic stream (506). If the datagram is not a fragment (i.e., not a fragment of a larger data packet), then reassembly may be automatic and extended throughout other layers of the network stack, such as layer 4 (508). If the datagram is a fragment of a larger data traffic stream, then fragment reassembly occurs, as described below in connection with ~~Figure 5B~~ Figures 6A and 6B (510).--

Please replace the paragraph beginning on page 12, line 18, with the following rewritten paragraph:

-- Figure 7 illustrates a process used in an embodiment to process overlapping fragments, as in step 612 of Figure 6B. In step [[712]] 702, the overlapping portions of the fragments are

compared to determine whether they are the same. If they are the same, the process of Figure 7 ends and the overlapping fragments are processed normally (i.e., the overlap is resolved in accordance with the configuration of the security system/application host. If there is a mismatch, i.e., one fragment has different data than the other for the overlapping portion, the process proceeds to step 704, in which expanded buffering is initiated. The expanded buffering is initiated to ensure that overlapping fragments are stored so as to be available for further processing, as described more fully below. In some embodiments, expanded buffering may be initiated as soon as overlapping fragments are detected, i.e., regardless of whether and/or before a determination is made regarding whether the overlapping portions do not match. In some embodiments, step 702 is omitted and the remaining steps of Figure 7 performed regardless of whether there is a mismatch. In step 706, an alert is generated. For example, an alert may be sent to a network security administrator indicating that overlapping fragments with mismatched data for the overlap portion have been detected, which as described herein could indicate an attempt to evade a security system or application. In step 708, the security system or application determines the configuration of the destination (target) host associated with the flow to which the overlapping fragments pertain. Specifically, the security system or application determines how the target host is configured to reassemble overlapping fragments (e.g., whether it uses for the overlap portion the data from the first-received fragment or instead the data from the later-received fragment). In some embodiments, step 708 may comprise querying an information base. In some embodiments, step 708 may comprise querying the destination (target) host. In step 710, the overlapping fragments are reassembled in accordance with the configuration of the destination (target) host. This allows the security system or application to see the data flow as it actually is reassembled by and appears to the target host. In this way, when the data is compared

to known attack signatures, e.g., an attack is detected if the data as reassembled by the target host would comprise an attack.--